GUIA 12
EJERCICIO 1

```
[Serializable]
public class Vehiculo : IComparable {

    private string nroPatente;
    private Cliente dueño;

    public Vehiculo (string patente, Cliente dueño){

        this.nroPatente = patente;
        this.dueño = dueño;
    }

    public Cliente VerDueño(){

    return dueño;
    }

    public string VerPatente(){

    return patente;
    }

    public int CompareTo(object obj){

        Vehiculo nVehiculo = obj as Vehiculo;
        if(nVehiculo != null){

            return this.nroPatente.CompareTo(nVehiculo.nroPatente);
        }
        return -1;
    }

    public override string ToString(){

        return $"{this.nroPatente}";
    }

}
```

```
[Serializable]
public class abstract Ticket {

    protected int nroOrden;
    private DateTime FechaHora;

    public Ticket () {
        FechaHora = DateTime.Now;
    }

    public int VerNro() {
        return nroOrden;
    }

    public DateTime VerFechaHora() {
        return FechaHora;
    }

    public override string ToString() {
        return $"{this.nroOrden}; {this.FechaHora:"dd-MM-xxxx HH:MM"}";
    }
}

[Serializable]
public class Denuncia : Ticket {

    private Vehiculo dominio;
    private static int numero;

    public Denuncia (Vehiculo asegurado) {
        this.dominio = asegurado;
        base.nroOrden = numero++;
    }

    public override string ToString() {
        return $"DENUNCIA; {base.ToString}; {this.dominio.ToString()}";
    }
}
```

```
[Serializable]
public class Cliente : Ticket{

    private long dni;                    public int TipoVehiculo { get; set; }
    private static int numero;

    public Cliente (string dni) {

        Foreach (char c in dni) {
            if (!char.IsDigit(c)) {
                throw new DNINoValidoException("Hay caracteres no numéricos");
            }
        long valor = Convert.ToInt64(dni);
        if(valor < 5000.000) {
            throw new DniNoValidoException("DNI debe ser mayor a 5.000.000");
        }
        this.dni = valor;
        base.nroOrden = numero++;
    }

    public override string ToString() {
        return $"NuevoCliente; {base.ToString()}; {this.dni}; {this.TipoVehiculo}";

                //NuevoCliente; nroOrden; FechaHora; dni; TipoVehiculo
    }
```

```
[Serializable]
public class Agencia {
    private List<Vehiculo> listaVehiculos = new List<Vehiculo>{
        new Vehiculo("LZL555", new Cliente("38523469"),
        new Vehiculo("ABC123", new Cliente("45888623"),
        new Vehiculo("JJJ555", new Cliente("40582321"),
        new Vehiculo("ASD456", new Cliente("37254456"),
        new Vehiculo("BBB123", new Cliente("43252728")
    };

    private Queue<Denuncia> denuncias = new Queue<Denuncia>();
    private Queue<Cliente> nuevosClientes = new Queue<Cliente>();
    public List<Ticket> ListaAtendidos { get; set; } = new List<Ticket>();
    public void AgregarTicket(Ticket turno) {
        if(turno != null) {
            if(turno is Denuncia) {
                denuncias.Enqueue(turno as Denuncia);
            }
            else
            if(turno is Cliente) {
                nuevosClientes.Enqueue(turno as Cliente);
            }
        }
    }
    public Ticket AtenderTicket(int tipo) {
        Ticket ticket = null;
        if(tipo == 1) {
            if(denuncias.Count > 0) {
                ticket = denuncias.Dequeue;
            }
        else
        if(tipo == 2) {
            if(nuevosClientes.Count > 0) {
                ticket = nuevosClientes.Dequeue();
            }
        }
        if(ticket != null) { ListaAtendidos.Add(ticket); }
    return Ticket;
```

```
public Vehiculo BuscarVehiculo(string patente){
    Vehiculo v = new Vehiculo(patente, null);
    listaVehiculos.Sort();
    int idx = listaVehiculos.BinarySearch(v);
    if(idx > -1){
        return listaVehiculos[idx];
    }
    return null;
}

public void AgregarVehiculo(string patente, string dni){
    Vehiculo vehiculo = BuscarVehiculo(patente);
    if(vehiculo == null){
        Cliente cliente = new Cliente(dni);
        vehiculo = new Vehiculo(patente, dni);
        listaAutos.Add(vehiculo);
    }
}
}
```

④
```
public class DniNoValidoException : ApplicationException {
    public DniNoValidoException(): base(){}
    public DniNoValidoException(string message): base(message){}
    public DniNoValidoException(string message, Exception innerExcep): base(message, InnerException){}
}
```

```
public class FormPrincipal {
    Agencia agencia = new Agencia();
    public btnTicket_Click(){

        try{
            Ticket ticket = null;
            if(rbDenuncia.Checked){
            String patente = tbPatente.Text;
            Vehiculo vehiculo = agencia.BuscarVehiculo(patente);
            if(vehiculo != null){
                ticket = new Denuncia(vehiculo);
                tbPatente.Clear();
            }else{
                MessageBox.Show("El vehiculo con esa patente no existe. Debe registrarse
                como nuevo cliente", "ERROR", MessageBoxButtons.Ok, MessageBoxIcon.
                                                                    Error);
            }
        else
            if(rbNuevoCliente.Checked){
            string dni = tbDNI.text;
            ticket = new Cliente(dni)
            (Cliente)ticket.TipoVehiculo = cmbTipo.SelectedIndex + 1;

            if(ticket != null){
                agencia.AgregarTicket(ticket);
                lsbTurnos.Items.Add(ticket);
                tbDNI.Clear();
            }else{
                MessageBox.Show("Debes seleccionar una opcion para generar un Ticket");
            }

        }catch(DNINoValidoException ex){
            MessageBox.Show(ex.Message,"ERROR DNI",MessageBoxButtons.Ok, MessageBoxIcon.Error);
        }catch(Exception ex){
            MessageBox.Show(ex.Message, "ERROR", MessageBoxButtons.Ok, MessageBoxIcon.Error);
    }
}
```

```
public void btnAtenderDenuncia(_) {

    ticket = agencia.AtenderTicket(1);
    if (ticket != null) {
        lsbTurnos.Items.Remove(ticket);
        MessageBox.Show("Ticket Atendido (Denuncia)");
    }
    else {
        MessageBox.Show("No hay tickets de tipo Denuncia por atender");
    }
}

public void btnAtenderNuevoCliente_Click(_) {
    ticket = agencia.AtenderTicket(2);
    if (ticket != null) {
        lsbTurnos.Items.Remove(ticket);
        MessageBox.Show("Ticket Atendido (Nuevo Cliente)");
    }
    else {
        MessageBox.Show("No hay tickets de tipo 'Nuevo Cliente' por atender");
    }
}
```

```
public void btnImportarVehiculos_Click(...){

    OpenFileDialog ofd = new OpenFileDialog();
    ofd.Filter = "(csv)|*.csv";
    ofd.Title = "IMPORTACIÓN DE VEHICULOS";
    if (ofd.ShowDialog() == DialogResult.Ok){

        string path = ofd.FileName;
        FileStream fs = null;
        StreamReader sr = null;

        try{

            fs = new FileStream(path, FileMode.Open, FileAccess.Read);
            sr = new StreamReader(fs);
            sr.ReadLine(); //Encabezado (nro Patente;dni)

            while(!sr.EndOfStream){

                string linea = sr.ReadLine();

                string[] splitResult = linea.Split(';');

                string patente = splitResult[0].Trim();
                string dni = splitResult[1].Trim();

                Vehiculo vehiculo = new Vehiculo(patente, new Cliente(dni));

                agencia.AgregarVehiculo(patente, dni);
            }
        }catch(Exception ex){

            MessageBox.Show(ex.Message, "ERROR EN LA IMPORTACIÓN DE VEHICULOS", MessageBoxButtons.Ok,
                            MessageBoxIcon.Error);
        }

        finally{

            if(sr != null) sr.Close();
            if(fs != null) fs.Close();
        }

}
```

```
public void btnExportarTickets_Click(~) {

    SaveFileDialog sfd = new SaveFileDialog();

    sfd.Filter = "(csv)|*.csv";
    sfd.Title = "EXPORTACIÓN DE TICKETS";

    if(sfd.ShowDialog == DialogResult.OK){

        FileStream FS = null;
        StreamWriter sw = null;
        string path = sfd.FileName;
        try {

            FS = new FileStream(path, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(FS);

            sw.WriteLine("DENUNCIA;nROORDEN;FechaHora;Patente");
            sw.WriteLine("NUEVOCLIENTE;nROORDEN;FechaHora;DNI;TipoVehiculo");

            foreach(Ticket ticket in Agencia.ListaAtendidos){

                sw.WriteLine(ticket.ToString());
            }

        } catch(Exception ex){

            MessageBox.Show(ex.Message,"ERROR EN LA EXPORTACIÓN DE TICKETS", MessageBox-
                    Buttons.Ok, MessageBoxIcon.Error");
        }
        finally{

            if(sw != null) sw.Close();
            if(FS != null) FS.Close();
        }

}
```

```
private void FormPrincipal_Load(...){
    FileStream fs = null;
    try{
        fs = new FileStream("datos.bin", FileMode.Open, FileAccess.Read);
        BinaryFormatter bf = new BinaryFormatter();
        agencia = bf.Deserialize(fs) as Agencia;
    }catch(Exception ex){
        MessageBox.Show(ex.Message, "ERROR AL DESSERIALIZAR EL ARCHIVO", MessageBoxButtons.Ok,
            MessageBoxIcon.Error);
    }finally{
        if(fs != null) fs.Close();
    }
}

private void FormPrincipal_Closing(...){
    FileStream fs = null;
    try{
        fs = new FileStream("datos.bin", FileMode.Create, FileAccess.Write);
        BinaryFormatter bf = new BinaryFormatter();
        bf.Serialize(fs, agencia);
    }catch(Exception ex){
        MessageBox.Show(ex.Message, "ERROR AL SERIALIZAR EL ARCHIVO", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
    finally{
        if(fs != null) fs.Close();
    }
}
}
```